新連載

FPGAで学ぶVerilog HDL



第1回 プログラマブル・デバイスで論理設計を学ぼう

デバイスの記事　ビギナーズ

井倉将実

本連載では，ハードウェア記述言語（HDL）であるVerilog HDLによるLSI設計技術の基礎を解説する．プログラマブル・デバイスを搭載するボードで動作する回路を記述し，実際に動作させながら，Verilog HDLによる設計を身に付けていく．今回は，道具として活用するプログラマブル・デバイスについて説明した後，最初の簡単な回路を設計する．　（編集部）

筆者は子どものころにロボットのプラモデルをよく購入しました．そして，設定資料が掲載されている書籍を参照し，「このロボットのどこにどんな機能があるんだろう」と想像しながら組み立てていくのが楽しみでした．気に入ったものは何度も買い，購入回数が10回を超えたものもありました．エンジンやコックピットを想像してはワクワクしたものです．オモチャ好きだった父親から「中に何が詰まっているのか，ということを考えながら組み立ててみなさい」と教わりました．

時は流れ，しごとでハードウェア開発を行うようになりました．初めて開発するものについては，最初は内部の詳細な構造がわかりません．しかし，「アレとアレを組み合わせればできそうだ」というひらめきと，それまでの経験によって，製品を作り上げています．プラモデルの場合とは事情もやっていることも違うのですが，「作り上げるためにはどのようなものが必要なのか」ということを想像し，足りない部分は資料などを参考にして作り上げていく点は同じなのだと思います．

1. LSI設計技術を身に付ける

ハードウェア設計では，回路図が広く用いられています．しかし，最近のディジタルLSI（large scale integrated circuit；大規模集積回路）設計では，ハードウェア記述言語（HDL：hardware description language）を使用するのが一般的です．

HDLにはいくつかの種類があります．例えば，ソフトウェア開発で広く用いられているC言語やJavaをベースとした言語を使うこともあります．しかし，LSI設計の現場で広く用いられているのはVerilog HDLとVHDLです．

● 言語設計の考えかた

複数の方式が普及し始めると，どちらが良いかという議論がかならず出てきます．LSI設計においても，Verilog HDLが良いか，VHDLが良いかとよく議論されています．

Verilog HDLでもVHDLでも，LSIを設計するのに問題なく利用できます．したがって，多少でも言語による設計の経験があるのであれば，その言語で設計すればよいと筆者は考えます．

ここで，言語で記述した回路を実際の回路に変換するための開発ツールのことを忘れてはなりません．開発ツールには高価なものが多く，技術サポートや協力会社との連携の問題などから，企業では設計手法や使用する開発ツールが決められているのが一般的です．そして，ディジタル民生機器などで使われるASIC（application specific integrated circuit；特定用途向けIC）と呼ばれるLSIの設計では，Verilog HDLが広く用いられています．

ASIC設計を前提とするか，もしくはその可能性があるなら，Verilog HDLは必須の言語です．逆に言えば，今に限らず将来でも，ASIC設計を行う可能性がある場合や，将来LSIを設計してみたいが何を学べばよいか悩んでいる場合には，Verilog HDLの学習をオススメします．

Verilog HDLは，LSIの検証を効率良く行うために企業

によって開発された言語がベースとなっており，それがIEEE 1364として標準化されました．

LSIを言語で設計するということをイメージしにくい方もいるかもしれません．Verilog HDLもソフトウェアと同じように，LSIを動かすためのプログラミング手法の一つだと考えればよいでしょう．プログラムを実行する部分が「CPU」と呼ばれているかどうかという違いと考えれば，なにも難しいことはありません．

Verilog HDLによるハードウェアの開発は，比較的敷居が低く，ソース・コードも読みやすく，必要最低限の言語知識で始められます．

● Verilog HDL設計の学びかた

Verilog HDLによる設計技術を学習するにあたり，言語仕様書の隅から隅まですべて覚える必要はありません．実際に，言語仕様のうちのごく一部の知識があれば，ある程度の設計ができてしまいます．

ある目的を達成しようとしたとき，いくつかの方法が用意されていることが普通です．ですから，「どれが正攻法なのか」というのは，個人のスタイルにもよりますし，押し付けられるべきことでもありません．自分のペースで学習するのがいちばんです．

ただし，企業では「記述スタイル」や「記述ガイドライン」といった記述に対する作法が用意されていることがあります．これは守るようにしましょう．現実的には，何かしらの指針となる参考書を片手に，学習を進めていくのがよいでしょう．また，解説書を読むだけの机上の知識だけではなく，実機に触れながら学習することをお勧めします．

● 経験しながら技術を身に付ける

プラモデルは組み立てに失敗しても作り直すことができました．そして，何度も失敗しながら繰り返し組み立てることで，じょうずに作る技術を身に付けていったものです．

しかし製品開発では，限られた納期の中で不ぐあいを出さないようにしなければなりません．製品の市場への投入が遅れれば，商機を逃してしまいます．製品の多彩な機能を実現する心臓部のLSIは，回路規模が大きく設計難易度が格段に高くなります．1億円を超える製造費がかかる場合もあり，何度も作り直すことなどできません．販売後にトラブルが起きてしまったら，ばくだいな回収費用がかかってしまうかもしれません．

それでは，LSI設計技術を身に付けるにはどうしたらよいのでしょうか．プラモデルを組み立てるように，何度も繰り返しながら経験を積み，ときには失敗しつつ，技術を身に付けることはできないのでしょうか．

そこで，注目するのがプログラマブル・デバイスというLSIです．

2．プログラマブル・デバイスについて

プログラマブル・デバイスとは，内部の回路構成を設計者が手元で決めることができ，多くの場合，再設計も可能なLSIです．PLD（programmable logic device）とも呼ばれます．

内部構造や特徴から，FPGA（field programmable gate array），CPLD（complex programmable logic device），SPLD（simple programmable logic device）などに分類されています（**図1**）．狭義には，CPLDとSPLDをPLDと呼びます．

● FPGAの特徴

FPGAは，多数の小規模な基本論理ブロックを，縦横無尽に張り巡らされた配線で相互接続した構造を採るのが一般的です（**図2**）．基本論理ブロックは，数本の入力信号から出力信号の論理を決める機能しか持ちません．したがって1個ではたいした機能を実現できないのですが，これを数千～数十万個組み合わせることで，大規模な回路を構成します．

数本の入力から出力信号を得るための論理を決めるのは，ルックアップ・テーブル（look-up table）と呼ばれるRAM

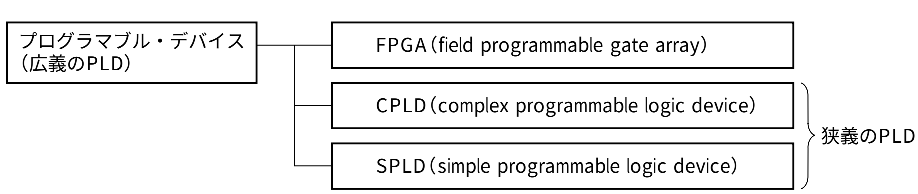
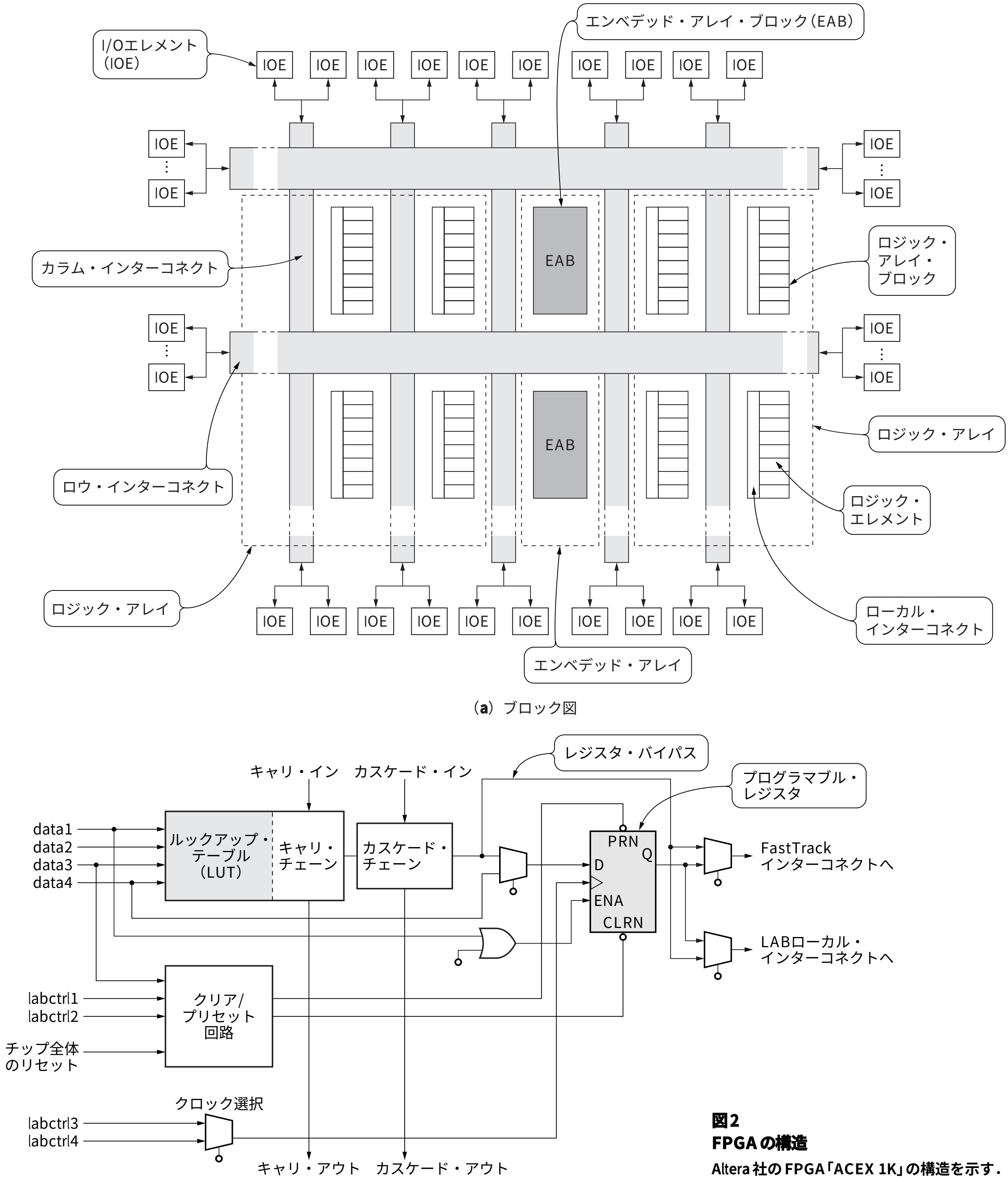


図1
プログラマブル・デバイスの分類
内部の回路構成を設計者が手元で決められる論理LSIには，大きく分けてFPGA，CPLD，SPLDの3種類がある．

(randam access memory)です．したがって，FPGAは一般的にSRAM(static RAM)を基本とした構造になります．つまり，電源を落とすと内部の情報が初期化されてしまいます．そこで，「コンフィグレーションROM(「ブートROM」とも呼ばれる)という不揮発性のメモリを付ける必要があります．電源を投入すると，外部のコンフィグレーションROMから回路データがFPGAに転送されます．そして，FPGAの内部回路構成が確定した後で，本来の動作を開始します．この時間は，回路規模によって異なります



(**a**) ブロック図

(**b**) 基本論理ブロック(ロジック・エレメント)の構造

図2
FPGAの構造

Altera社のFPGA「ACEX 1K」の構造を示す．多数の基本論理ブロック(ロジック・エレメント)と縦横の配線(インターコネクト)で構成されている．

が，ms（ミリ秒）から数sがかかります．また，コンフィグレーションROMを内蔵したFPGAもあります．

● CPLD/SPLDの特徴

CPLD/SPLDでは，「マクロ・セル」と呼ばれる比較的大規模な論理ゲートの集まりが基本単位になります．プログラマブル・デバイスが登場した当初のマクロ・セルを一つだけ搭載したものをSPLD，複数のマクロ・セルが集積され，さらに互いに接続された構造を採るものをCPLDと呼んでいます（**図3**）．

一つのマクロ・セルは，10本を超える入力信号と複数の出力信号を持ち（例えば16入力8出力），任意の論理構造を実現するための論理ゲートの集まりで構成されます．一つのマクロ・セルだけでもある程度の回路を実現できます．また，複数のマクロ・セルを使って，より規模の大きな設計に対応することもできます．

CPLDでは，内部の回路構成を決定する素子にEEP-ROM（electrically erasable and programmable read only memory）が使われるのが一般的です．このメモリは，CPLDの内部にあらかじめ用意されています．不揮発性のため，電源投入直後から動作します．

● FPGAとCPLD/SPLDの比較

CPLDは中心にある太い配線領域（PIA：Program-

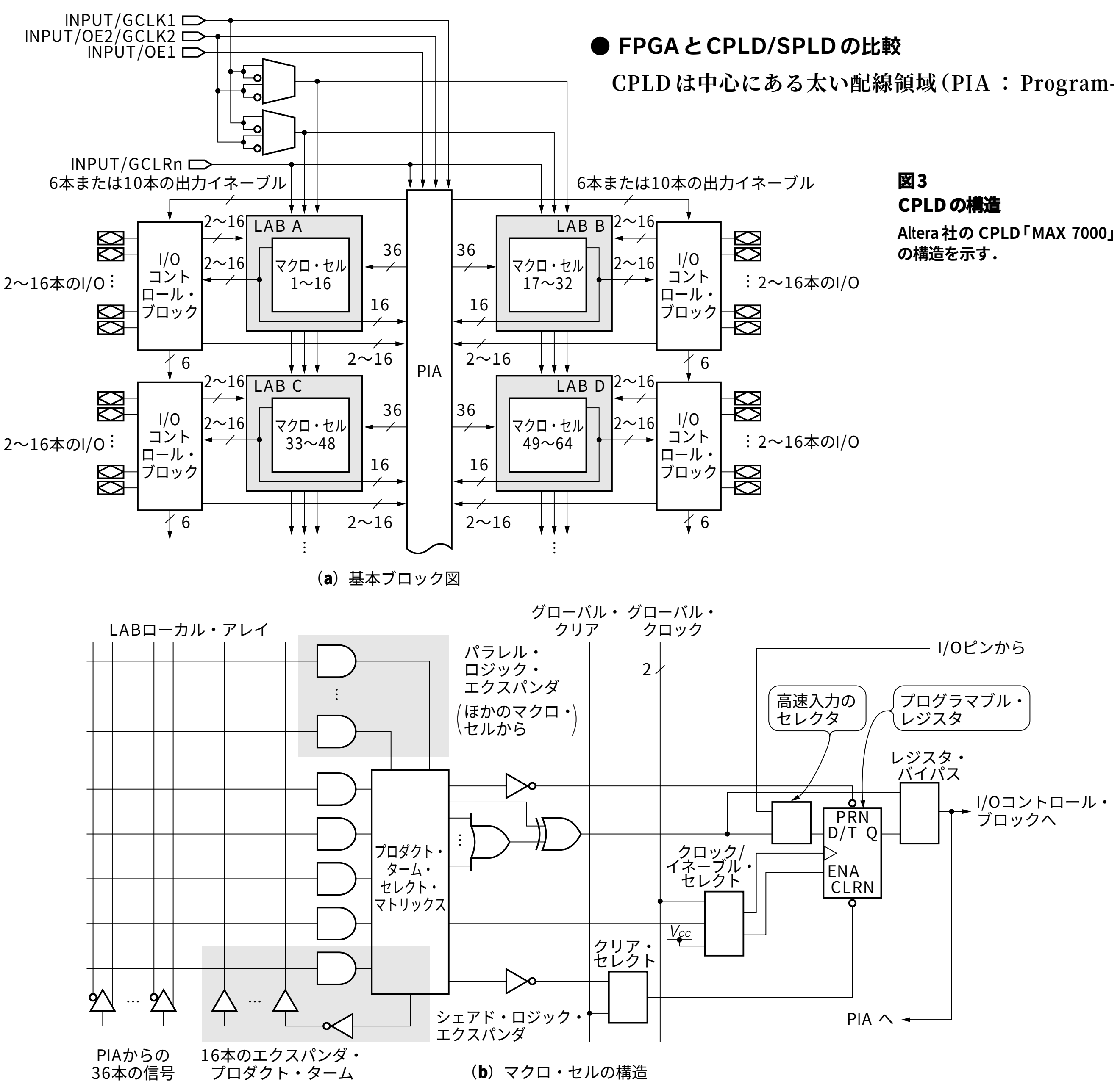


図3 CPLDの構造
Altera社のCPLD「MAX 7000」の構造を示す．



mable Interconnect Array）でマクロ・セルどうしを接続します．これに対して，FPGAでは基本論理ブロックの入出力は格子状の配線領域につながっています．一般には，CPLDよりもFPGAのほうが設計の自由度が高くなります．

また，信号を入力してから出力を得るまでの「遅延時間」についても，CPLDとFPGAには明確な違いがあります（**図4**）．CPLDでは，入出力ピンとマクロ・セルの入出力は1対1で接続されます．マクロ・セル間をつなぐ配線も決まっています．このため，信号の入力から出力までの信号経路が限られ，遅延時間を予測しやすくなっています．タイミング要求の厳しい設計ならCPLDが便利です．これに対してFPGAは，多数の基本論理ブロックのうち，設計した回路をどのブロックに配置し，どの配線でつなぐかを自由に決めることができます．多くの場合は設計ツールが自動的に行いますが，回路全体の状況によって配置や配線が変わってきます．配線経路が大きく変わるということは，遅延時間が変わるということであり，遅延時間を予測しづらいということになります．

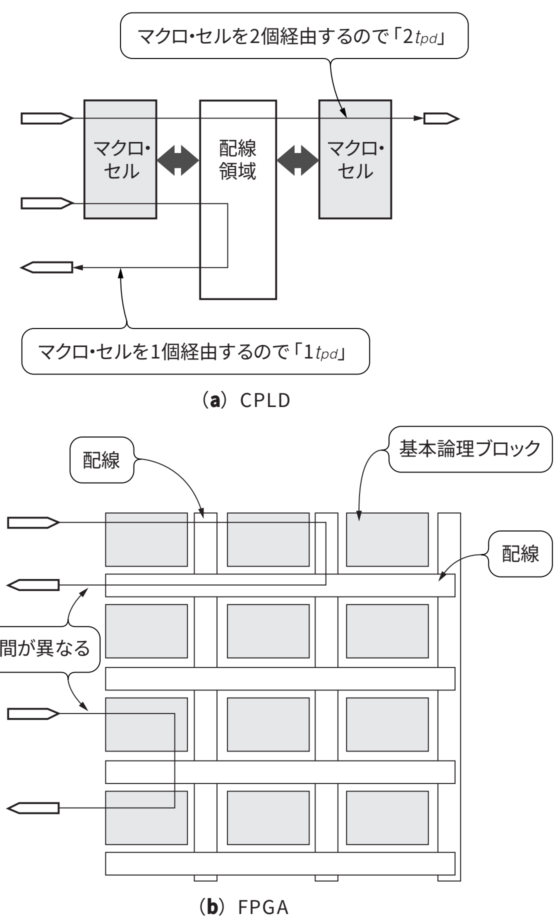


図4　CPLDとFPGAの違い
CPLDでは，信号の入力から出力までの信号経路が限られているので遅延時間を予測しやすい．FPGAは同じ回路であっても配置配線結果によって遅延時間が変化する．

3. LSI設計の流れ

ここではLSIの設計手順について説明します．概略の流れを**図5**に示します．

● 仕様設計

仕様設計工程では，作りたいものを考え，どのような機能が必要であるかを検討します．作りたいもののカタログやデータシートを作成する作業だと考えればイメージしやすいと思います．

この段階で，すべての機能を洗い出しておくことが理想です．小規模な回路であれば仕様ができ上がった時点でほぼ回路の動きやタイミングなどの詳細が決まってしまうため，この作業は非常に重要です．

● 機能設計

機能設計工程では，作成した仕様をもとに，回路の実現方式を検討します．ここでたいせつなことは，仕様を実現するために必要な回路構成はどのようなものかということを見極めることです．

しかし，ある程度の設計経験がないと，どのような機能が必要なのかということが見えてこないものです．このようなときには，まずはおおざっぱに回路全体をいくつかの機能ブロックに分けてしまいます．そしておのおのの機能ブロックの中身を後から検討していくとよいでしょう（トップダウン設計）．一つの機能ブロックが大きい場合には，さらに小さな機能ブロックに分割して考えていきます．

必要な機能ブロックをすべて開発するのか，または過去の設計資産を活用するのかについても考えます．設計資産はIP（intellectual property）コアと呼ばれています．すで

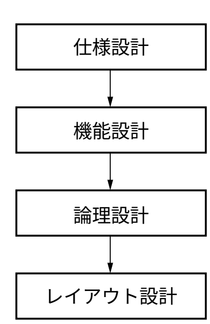


図5
LSI設計の流れ
LSI設計工程をシンプルにまとめた．検証工程については省略している．HDL設計を学ぶ段階では，FPGAに回路を実装して動作させてみるほうが楽しい．ただし，実際の設計業務では検証のほうが重要になってきている．

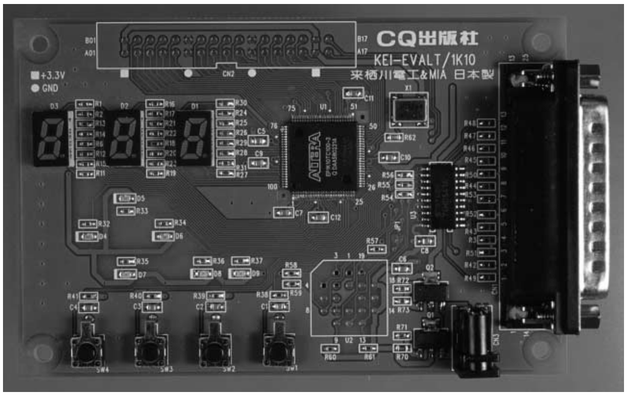


写真1 使用するFPGAボード

「FPGAボードで学ぶ論理回路設計」に付属するFPGAボード．FPGAは，Altera社のEP1K10（1万ゲート相当）である．周辺回路として，スイッチとLEDを搭載している．FPGAに回路データを書き込むための回路を搭載しているので，パソコンのプリンタ・ポートに接続するだけで利用できる．

に完成し，使用実績のある機能ブロックを活用することで，設計期間の短縮や信頼性の向上などを図ることができます．

IPコアは，そのまま使うことができないケースがほとんどですが，複雑な機能を短い期間で実現できれば，開発期間の短縮に大きく寄与します．ノウハウを持つIPコア・ベンダの協力を得られることも大きな利点です．ただし，市販のIPコアには高価なものもあり，数百万円から場合によっては数億円の費用負担を強いられることもあります．自分たちで設計するところとIPコアを活用するところを開発費用の面からも検討する必要があります．

● 論理設計，レイアウト設計

論理設計工程では，HDLを使って機能ブロックを記述していきます．それをパソコンやワークステーションで動作する開発ツールを使って実際の回路に変換（論理合成）します．そして，レイアウト設計工程で配置配線を行います．

FPGA向けの設計であれば，論理設計とレイアウト設計を一つのツールの中でまとめて行うことができます．

ASIC設計では，設計した回路が正しく動作するかどうかを徹底的に検証し，問題がないと確信してからデータを製造工場に渡すことになります．ASICの検証でも，FPGAを用いるのが普通です．

4. スイッチでLEDをON/OFFする回路の設計

ここでは，Verilog HDLで簡単な回路を記述してみることによって，記述にあたっての大原則を学びます．また，

表1 スイッチでLEDを点滅させる回路の動作

スイッチの状態	入力レベル	LED動作	出力レベル
押す	L	点灯	L
離す	H	消灯	H

リスト1 スイッチでLEDを点滅させる回路のVerilog HDL記述

```
module LEDSW ( SW_in, LED_out ) ;
 input SW_in ;
 output LED_out ;

 wire  LED_out ;

assign LED_out = SW_in ;

endmodule
```

FPGAに実装して動作させてみます．

設計する回路は，FPGAに接続されたスイッチの状態に応じて，LEDを点灯・消灯させる回路です．ここでは，CQ出版社のボード付き書籍「FPGAボードで学ぶ論理回路設計」[(1)]に付属するFPGAボード（**写真1**）を用いますが，スイッチとLEDが接続されたFPGA評価ボードであれば，ほかのものも利用可能です．設計ツールは，米国Altera社の「Quartus II Web Edition」です．ツールの使いかたは，ここでは説明しません．多くの文献があるので，それらを参照してください[(1), (2)]．

回路を設計する場合，まず入力条件と出力条件をまとめるのが一般的です．今回の回路の入出力条件と動作を**表1**にまとめます．この動作を実現するVerilog HDL記述を**リスト1**に示します．

● モジュール宣言

Verilog HDLでは，ひとまとまりの機能を「モジュール」として扱います（**図6**）．モジュールは，moduleで名まえを付けて宣言します．endmoduleはmoduleで宣言された機能の終わりを表します．

モジュール宣言では，入出力に使われる信号名のリスト（ポート・リスト）を記述します．**リスト1**では，SW_inとLED_outという二つの信号名があります．

信号名はAでもBでもかまいませんが，機能や用途が想像できるようにすると，ミスのないわかりやすい記述になります．ここでは，SW_in（スイッチからの入力），LED_out（LEDへの出力）と命名しています．Verilog HDLの規則では，英字またはアンダ・スコア（_）で始まり，英数字，アンダ・スコア，ドル（$）による文字列を使うことができます．Verilog HDLでは大文字と小文字を区別するので注意が必要です．

```
module モジュール名 （ ポート・リスト ） ;
 input   入力信号名 ;
 output  出力信号名 ;
 inout   双方向信号名 ;

 wire   組み合わせ回路の信号 ;
 reg    順路回路の信号 ;

動作記述部

endmodule
```

図6　Verilog HDLによる記述の基本

ひとまとまりの機能を「モジュール」として扱う．外部入出力信号を定義するポート宣言部，内部信号を定義するネット/レジスタ宣言部，実際の回路の動作を記述する動作記述部がある．

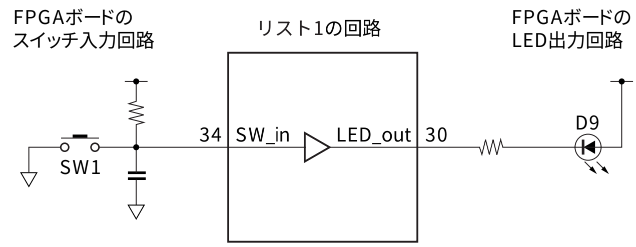


図7　スイッチでLEDを点滅させる回路のイメージ

スイッチ（SW1）を押すと，SW-inに“L”が入力される．それがそのままLED-outに出力され，LEDが点灯する．

ポート・リストの信号名の区切りには，カンマ（,）を使います．最後に宣言した信号名の後にはカンマは不要です．

moduleの最後にはセミコロン（;）を付けます．Verilog HDLでは，文の終わりにはセミコロンを付けます．

● ポート宣言

モジュール宣言では信号名を記述していますが，その信号の方向については指定していません．そこで，input（入力），output（出力），inout（双方向）を使って，信号の方向を指定します．

リスト1では，SW_inは入力信号，LED_outは出力信号と宣言しています．

● ネット/レジスタ宣言

論理回路は，ANDゲートやORゲートなどの組み合わせで表現され，入力信号の変化とともに即座に出力が変化するもの（組み合わせ回路）と，クロック信号などのタイミングで変化するもの（順序回路）の2種類に大別できます．そこで，各出力信号がどちらのタイプになるのかを明示する必要があります．Verilog HDL設計でもっとも多く使われるのは，組み合わせ回路であることを示すwireと順序回路であることを示すregの2種類です．

リスト1では，入力信号の変化によってLEDの状態を変化させるので，wireで宣言しています．

● 動作記述部

assign文は，信号の代入を行う記述です．右辺の信号を，左辺の信号に代入します．

```
assign LED_out = SW_in ;
```

は，SW_in信号の値をLED_outに伝えることを表しています．つまり，スイッチの入力をLEDの出力に伝えるのです．この記述により，**表1**で表した動作を満たすことができます．

なお，assign文の左辺で取り扱うことができるのは，wireで定義された信号だけです．

● 動作確認

写真1のFPGAボードを利用する場合，Altera社のQuartus IIを使って論理合成と配置配線を行います．このとき，SW_inを34番ピン，LED_outを30番ピンに割り当てます．FPGAボードに回路をダウンロードすると，SW1を押すとLEDのD9が点灯，離すと消灯する動作を確認できます．

こうしてでき上がった回路のイメージを，**図7**に示します．

参考・引用*文献

(1) 山際伸一；FPGAボードで学ぶ論理回路設計，CQ出版社，2003年5月．
(2) Design Wave Magazine編集部 編；FPGA/PLD設計スタートアップ 2005-2006年版，CQ出版社，2005年5月．
(3) 小林優；改訂 入門Verilog HDL記述，CQ出版社，2004年6月．
(4) Design Wave Magazine編集部 編；FPGA活用チュートリアル，CQ出版社，2004年3月．
(5) **ACEX 1K Programmable Logic Device Family*，Altera，May 2003.
(6) **MAX 7000A Programmable Logic Device*，Altera，September 2003.

いくら・まさみ
FPGA-Magician

<筆者プロフィール>

井倉将実．サムイ島からフィリピン海を見下ろしながら執筆しました．入門者向けにかなり注意して執筆したつもりですが，いかがだったでしょうか．ご意見をいただけると幸いです．